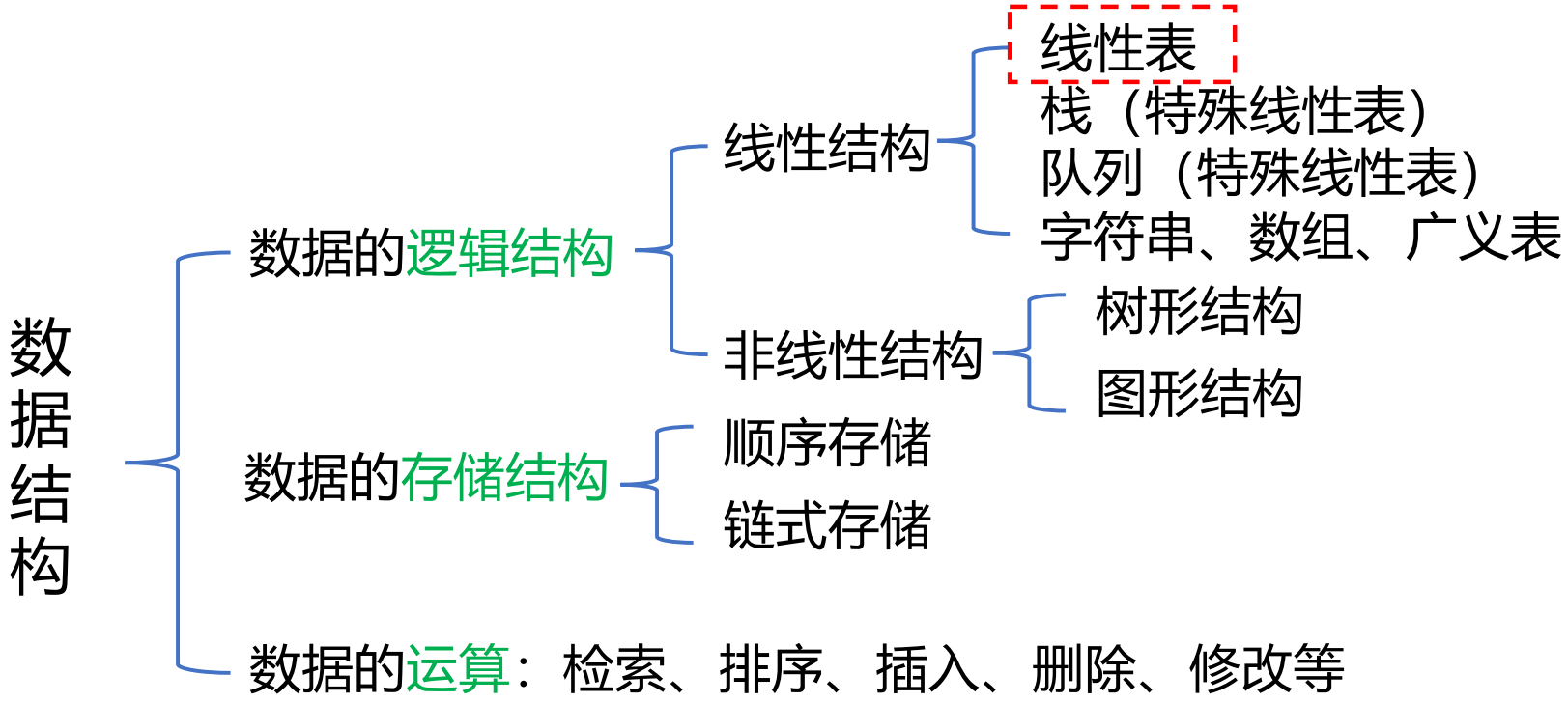
知识回顾


数据结构
数据的逻辑结构
线性结构
线性表
栈（特殊线性表）
队列（特殊线性表）
字符串、数组、广义表
非线性结构
树形结构
图形结构
数据的存储结构
顺序存储
链式存储
数据的运算：检索、排序、插入、删除、修改等

第2章 线性表

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

2.1 线性表的类型定义

2.2 线性表的顺序表示和实现

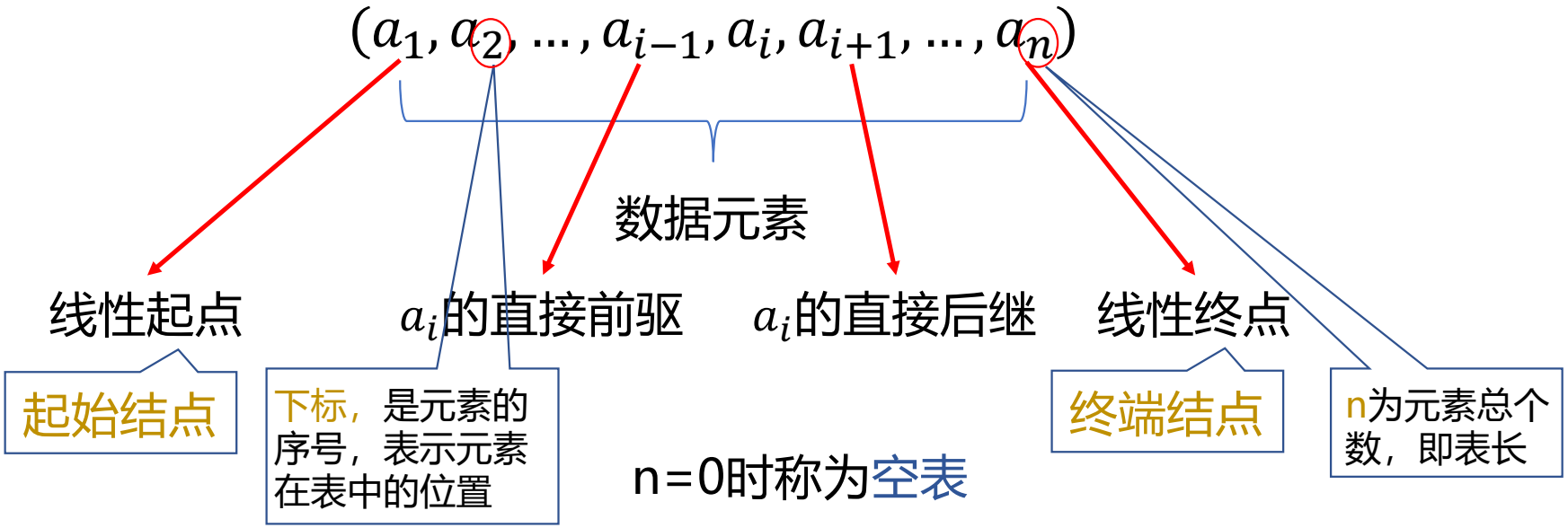
2.3 线性表的链式表示和实现

2.4 顺序表和链表的比较

2.5 线性表的应用

2.1 线性表的类型定义

线性表是具有相同特性的数据元素的一个有限序列



2.1 线性表的类型定义

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 线性表（Linear List）：

  由n(n≥0)个数据元素(结点)$a_1, a_2, \ldots, a_n$组成的有限序列。

  - 其中数据元素的个数n定义为表的**长度**
  - 当n=0时称为**空表**
  - 将非空的线性表(n > 0)记作: $(a_1, a_2, \ldots, a_n)$
  - 这里的数据元素$a_i(1 \le i \le n)$只是一个抽象的符号，其具体含义在不同的情况下可以不同。

线性表的例子

- *例1：* 分析26个英文字母组成的英文表

$$(\mathrm{A}, B, C, D, \ldots\ldots, Z)$$

数据元素都是字母；元素间关系是线性

- *例2：* 分析学生情况登记表

学号	姓名	性别	籍贯	专业
42411	张三	男	浙江	计算机科学与技术
42412	李四	女	上海	计算机科学与技术
42413	王五	男	江苏	计算机科学与技术
:	:	:	:	:

数据元素都是记录；元素间关系是线性

线性表的例子

- *例1：* 分析26个英文字母组成的英文表

$$(\mathrm{A}, B, C, D, \ldots\ldots, Z)$$

数据元素都是字母；元素间关系是线性

- *例2：* 分析学生情况登记表
- *例3：* 某单位历年拥有计算机的数量（6，17，28，50，92，188）
- *例4：* 12星座（白羊座、金牛座、双子座、巨蟹座、狮子座、处女座、天秤座、天蝎座、射手座、摩羯座、水瓶座、双鱼座）

同一线性表中的元素必定具有相同特性，数据元素间的关系是线性关系

线性表的逻辑特征

- 从以上例子可看出线性表的逻辑特征是：
  - 在非空的线性表，有且仅有一个开始结点a_1，它没有直接前趋，而仅有一个直接后继a_2；
  - 有且仅有一个终端结点a_n，它没有直接后继，而仅有一个直接前趋a_{n-1}；
  - 其余的内部结点$a_i(2 \leq i \leq n-1)$都有且仅有一个直接前趋a_{i-1}和一个直接后继a_{i+1}。

线性表是一种典型的线性结构

案例引入

【案例2.1】一元多项式的运算：实现两个多项式加、减、乘运算

$$P_n(x) = p_0 + p_1x + p_2x^2 + \cdots + p_nx^n$$

线性表$P = (p_0, p_1, p_2, \ldots, p_n)$

（每一项的指数i隐含在其系数p_i的序号中）

例如：$P(x) = 10 + 5x - 4x^2 + 3x^3 + 2x^4$

指数(下标i)	0	1	2	3	4
系数p[i]	10	5	-4	3	2

用数组来表示

案例2.1：一元多项式的运算

$R_n(x) = P_n(x) + Q_m(x)$

线性表$P = (p_0 + q_0, p_1 + q_1, p_2 + q_2, \cdots, p_m + q_m, p_{m+1}, \cdots, p_n)$

稀疏多项式
$S(x) = 1 + 3x^{10000} + 2x^{20000}$

将会造成存储空间很大的浪费，怎么办?

案例2.2：稀疏多项式的运算

多项式非零项的数组表示

(a) $A(x) = 7 + 3x + 9x^8 + 5x^{17}$

下标i	0	1	2	3
系数p[i]	7	3	9	5
指数	0	1	8	17

(b) $B(x) = 8 + 22x^7 - 9x^8$

下标i	0	1	2
系数p[i]	8	22	-9
指数	1	7	8

$$P_n(x) = p_1x^{e_1} + p_2x^{e_2} + \cdots + p_mx^{e_m}$$

线性表P=((p1,e1),(p2,e2),...,(pm,em))

线性表A=((7,0),(3,1),(9,8),(5,17))　线性表B=((8,1),(22,7),(-9,8))

2.1 线性表的类型定义

- 抽象数据类型线性表的定义如下：

ADT List{

数据对象： $D=\{a_i|a_i属于Elemset,(i=1,2,\dots,n,n\geq 0)\}$

数据关系： $R=\{<a_{i-1},a_i>|a_{i-1},a_i属于D,(i=2,3,\dots,n)\}$

基本操作：

InitList(&L); DestroyList(&L);

ListInsert(&L,i,e); ListDelete(&L,i,&e);

……等等

}**ADT** List

基本操作（一）

- **InitList(&L)** (Initialization List)
  - 操作结果: 构造一个空的线性表L。
- **DestroyList(&L)**
  - 初始条件: 线性表L已经存在。
  - 操作结果: 销毁线性表L。
- **ClearList(&L)**
  - 初始条件: 线性表L已经存在。
  - 操作结果: 将线性表L重置为空表。

基本操作（二）

- **ListEmpty(L)**
  - 初始条件: 线性表L已经存在。
  - 操作结果: 若线性表L为空表，则返回TRUE; 否则返回FALSE。
- **ListLength(L)**
  - 初始条件: 线性表L已经存在。
  - 操作结果:返回线性表L中的数据元素个数

基本操作（三）

- **GetElem(L,i,&e)**
  - 初始条件: 线性表L已经存在，1<=i<= ListLength(L)。
  - 操作结果: 用e返回线性表L中第i个数据元素的值。
- **LocateElem(L,e,compare())**
  - 初始条件: 线性表L已经存在，compare()是数据元素判定函数。
  - 操作结果: 返回L中第1个与e满足compare()的数据元素的位序。若这样的数据元素不存在则返回值为0。

基本操作（四）

- **PriorElem(L, cur_e, &pre_e)**
  - 初始条件: 线性表L已经存在。
  - 操作结果: 若cur_e是L的数据元素，且不是第一个，则用pre_ e返回它的前驱否则操作失败；pre_ e无意义。
- **NextElem(L, cur_e, &next_e)**
  - 初始条件: 线性表L已经存在。
  - 操作结果: 若cur_e是L的数据元素，且不是第最后个，则用next_e返回它的后继，否则操作失败，next_e无意义。

基本操作（五）

- **ListInsert(&L, i, e)**
  - 初始条件: 线性表L已经存在，1<=i<= ListLength(L)+1。
  - 操作结果: 在L的第i个位置之前插入新的数据元素e, L的长度加一。

插入元素e之前(长度为n)： $(a_1, a_2, \ldots, a_{i-1}, a_i, \ldots, a_n)$

插入元素e之后(长度为n+1)： $(a_1, a_2, \ldots, a_{i-1}, e, a_i, \ldots, a_n)$

基本操作（六）

- **ListDelete(&L,i,&e)**
  - 初始条件: 线性表L已经存在，1 <= i <= ListLength(L)。
  - 操作结果: 删除L的第i个数据元素，并用e返回其值，L的长度减一。
    - 删除前(长度为n): $(a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n)$
    - 删除后(长度为n-1): $(a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$
- **ListTraverse(&L, visited())**
  - 初始条件: 线性表L已经存在
  - 操作结果: 依次对线性表中每个元素调用visited()

2.1 线性表的类型定义

- 以上所提及的运算是逻辑结构上定义的运算。只要给出这些运算的功能是"做什么"，至于"如何做"等实现细节，只有待确定了存储结构之后才考虑。
- 后续课程中将学习线性表的存储及在存储结构上各操作的实现。
  - 2.2 线性表的**顺序**表示和实现
  - 2.3 线性表的**链式**表示和实现

第2章 线性表

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

2.1 线性表的类型定义

2.2 线性表的顺序表示和实现

2.3 线性表的链式表示和实现

2.4 顺序表和链表的比较

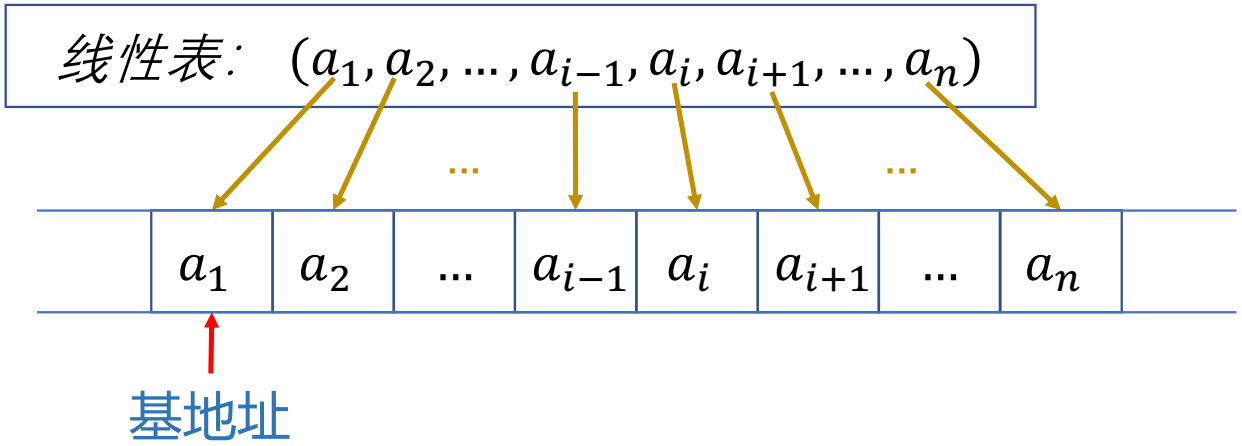
2.5 线性表的应用

2.6 案例分析与实现

2.2.1 线性表的顺序存储表示

线性表的顺序表示又称为顺序存储结构或顺序映像。

顺序存储定义：把逻辑上相邻的数据元素存储在物理上相邻的存储单元中的存储结构。



简言之，逻辑上相邻，物理上也相邻

线性表的第1个数据元素a_1的存储位置，称作线性表的起始位置或基地址。

顺序存储结构

例如: 线性表(1, 2, 3, 4, 5, 6)的存储结构:

1	2	3	4	5	6

依次存储，地址连续——
中间没有空出存储单元。

是一个典型的线性表顺序存储结构。

存储结构:

1	2			3	4	5	6

地址不连续——
中间存在空的存储单元。

不是一个线性表顺序存储结构。

线性表顺序存储结构占用一片连续的存储空间。知道某个元素的存储位置就可以计算其他元素的存储位置

顺序表中元素存储位置的计算

	a_1	a_2	…	a_{i-1}	a_i	a_{i+1}	…	a_n	

如果每个元素占用8个存储单元， a_i存储位置是2000单元，则a_{i+1}存储位置是?

2008单元

假设线性表的每个元素需占l个存储单元，则第$i+1$个数据元素的存储位置和第i个数据元素的存储位置之间满足关系:

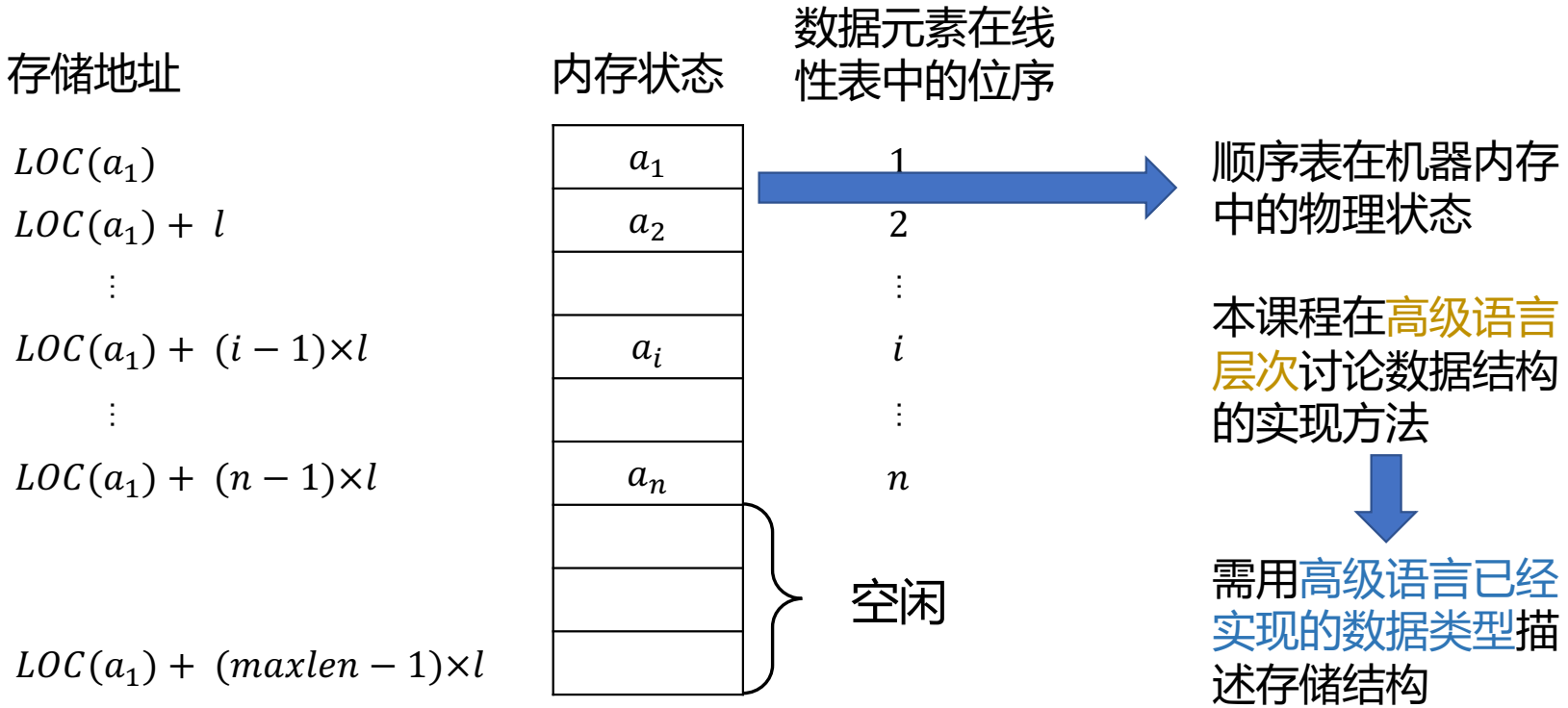
$$LOC(a_{i+1}) = LOC(a_i) + l$$

由此，所有数据元素的存储位置均可由第一个数据元素的存储位置得到:

$$LOC(a_i) = LOC(a_1) + (i-1)\times l$$

↑ 基地址

线性表顺序存储结构图示



顺序表在机器内存中的物理状态

本课程在高级语言层次讨论数据结构的实现方法

需用高级语言已经实现的数据类型描述存储结构

顺序表的特点: 以物理位置相邻表示逻辑关系。
任一元素均可随机存取。(优点)

2.2.1 线性表的顺序存储表示

顺序表（元素）{ 地址连续 依次存放 随机存取 类型相同 } 数组（元素）→ 用一维数组表示顺序表

线性表长可变（删除）

数组长度不可动态定义

一维数组的定义方式:

类型说明符　数组名[常量表达式]

说明: 常量表达式中可以包含常量和符号常量，不能包含变量。即C语言中不允许对数组的大小作动态定义。

2.2.1 线性表的顺序存储表示

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

顺序表（元素）{ 地址连续 依次存放 随机存取 类型相同 } 数组（元素）→ 用一维数组表示顺序表

线性表长可变（删除）
数组长度不可动态定义 } → 用一变量表示顺序表的长度属性

```
#define LIST_INIT_SIZE 100      // 线性表存储空间的初始分配量
typedef struct {
      ElemType elem[LIST_ INIT_ SIZE];
      int length;    // 当前长度
} SqList;
```

多项式的顺序存储结构类型定义

$$P_n(x) = p_1x^{e_1} + p_2x^{e_2} + \cdots + p_mx^{e_m}$$

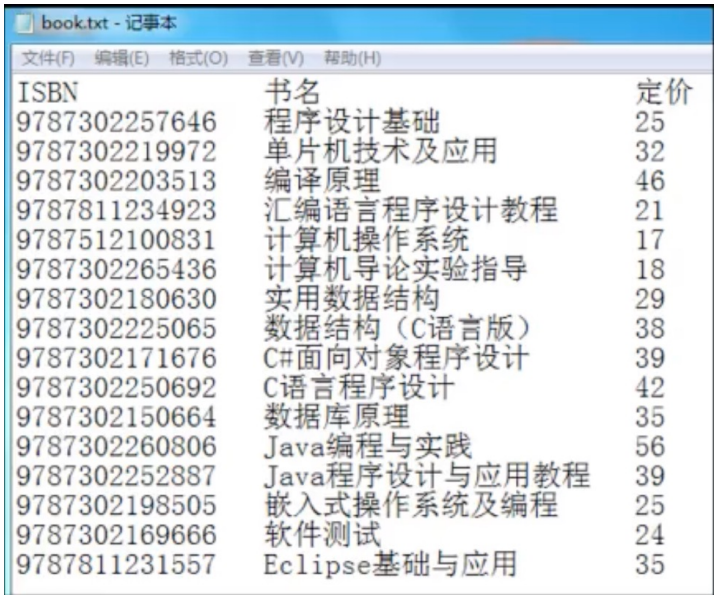
线性表P=((p1,e1),(p2,e2),...,(pm,em))

```
#define MAXSIZE 1000      //多项式可能达到的最大长度

typedef struct {          //多项式非零项的定义
    float p;              //系数
    int e;                //指数
} Polynomial;

typedef struct {
    Polynomial elem[MAXSIZE];
    int length;           //多项式中当前项的个数
} SqList;                 //多项式的顺序存储结构类型为SqList
```

图书表的顺序存储结构类型定义

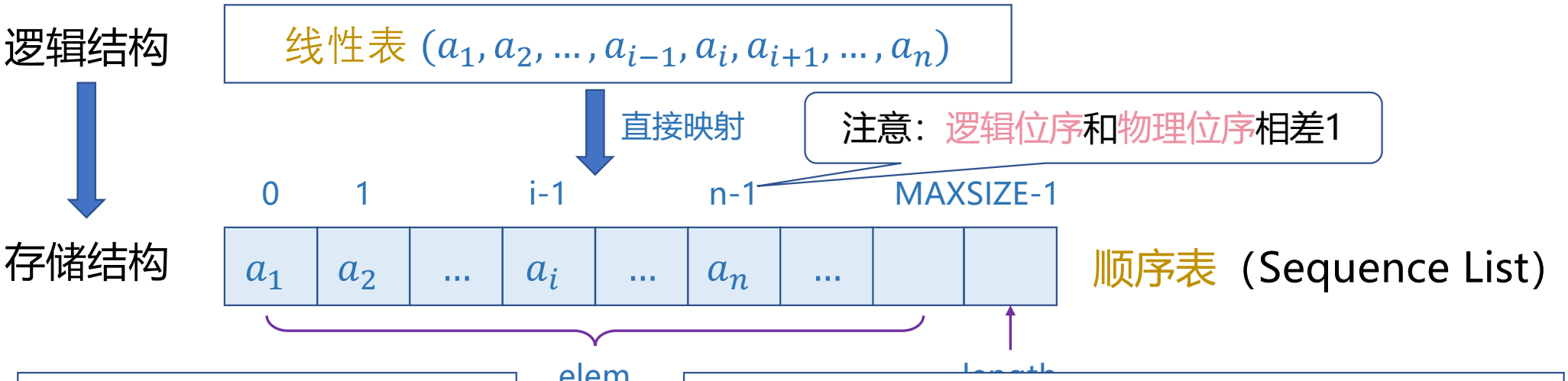



```
#define MAXSIZE 10000      //图书表可能达到的最大长度

typedef struct {           //图书信息定义
    char no[20];           //图书ISBN
    char name[50];         //图书名字
    float price;           //图书价格
} Book;

typedef struct {
    Book *elem;            //存储空间的基地址
    int length;            //图书表中当前图书个数
} SqList;                  //图书表的顺序存储结构类型为SqList
```

2.2.1 线性表的顺序存储表示




```
#define MAXSIZE 100
typedef struct {
    ElemType elem[MAXSIZE];
    int length;
} SqList;
```

```
typedef struct {
    ElemType *elem;
    int length;
} SqList;   //顺序表类型
L.elem=(ElemType*)malloc(sizeof(ElemType)*MAXSIZE);
```

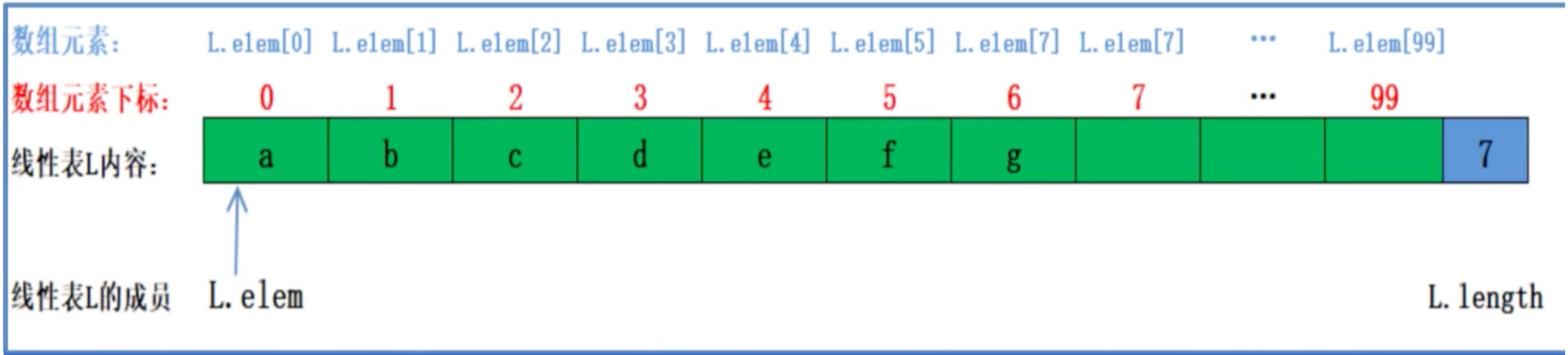
顺序表示意图


```
#define MAXSIZE 100
typedef struct {
        ElemType *elem;
        int length;
} SqList;   //定义顺序表类型
```

就像：

```
int a;    //定义变量a，a是int型
```

```
SqList L;   //定义变量L，L是SqList这种类型的，L是个顺序表
```



2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 线性表的基本操作：
  - InitList(&L)　　//初始化操作，建立一个空的线性表L
  - DestroyList(&L)　　//销毁已存在的线性表L
  - ClearList(&L)　　//将线性表清空
  - ListInsert(&L, i, e)　　//在线性表L中第i个位置插入新元素e
  - ListDelete(&L, i, &e)　　//删除线性表L中第i个位置元素，用e返回
  - IsEmpty(L)　　//若线性表为空，返回true，否则false
  - ListLength(L)　　//返回线性表L的元素个数
  - LocateElem(L, e)　//L中查找与给定值e相等的元素，若成功返回该元素在表中的序号，否则返回0
  - GetElem(L, i, &e)　　// 将线性表L中的第i个位置元素返回给e

补充:操作算法中用到的预定义常量和类型

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

```
//函数结果状态代码
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
#define OVERFLOW -2
//Status是函数的类型， 其值是函数结果状态代码
typedef int Status;
typedef char ElemType;
```

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【算法2.1】 线性表L的初始化(参数用引用)

```
Status InitList_Sq(SqList &L){                  //构造一个空的顺序表L
    L.elem=(ElemType*)malloc(sizeof(ElemType)*MAXSIZE); //为顺序表分配空间
    if(!L.elem) exit(OVERFLQW);                 //存储分配失败
    L.length=0;                                 //空表长度为0
    return OK;
}
```

2.2.2 顺序表基本操作的实现——补充：几个简单操作

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

销毁线性表L

```
void DestroyList(SqList &L) {
    if (L.elem) free(L.elem);   //释放存储空间
}
```

清空线性表L

```
void ClearList(SqList &L) {
    L.length = 0;   //将线性表的长度置为0
}
```

2.2.2 顺序表基本操作的实现——补充：几个简单操作

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

求线性表L的长度

```
int GetLength(SqList L) {
    return L.length;
}
```

判断线性表L是否为空

```
int IsEmpty(SqList L) {
    if (L.length == 0)   return 1;
    else return 0;
}
```

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【算法2.2】顺序表的取值(根据位置i获取相应位置数据元素的内容)

```
int GetElem(SqList L, int i, ElemType &e){
    if (i<1 || i>L.length) return ERROR;
                    //判断i值是否合理，若不合理，返回ERROR
    e=L.elem[i-1];  //第i-1的单元存储着第i个数据
    return OK;
}
```

随机存取

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 线性表的基本操作：
  - InitList(&L)　　//初始化操作，建立一个空的线性表L
  - DestroyList(&L)　　//销毁已存在的线性表L
  - ClearList(&L)　　//将线性表清空
  - ListInsert(&L, i, e)　　//在线性表L中第i个位置插入新元素e
  - ListDelete(&L, i, &e)　　//删除线性表L中第i个位置元素，用e返回
  - IsEmpty(L)　　//若线性表为空，返回true，否则false
  - ListLength(L)　　//返回线性表L的元素个数
  - LocateElem(L, e)　　//L中查找与给定值e相等的元素，若成功返回该元素在表中的序号，否则返回0
  - GetElem(L, i, &e)　　// 将线性表L中的第i个位置元素返回给e

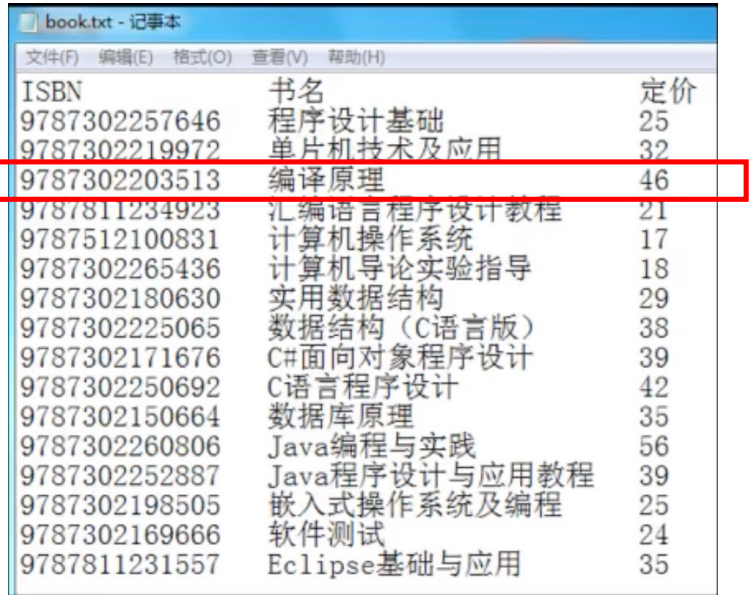
顺序表上的查找操作

按值查找

例如：在图书表中，按照给定书号进行查找，确定是否存在该图书

如果存在：输出是第几个元素

如果不存在：输出0



2.2.2 顺序表基本操作的实现

【算法2.3】顺序表的查找

- 在线性表L中查找与指定值e相同的数据元素的位置
- 从表的一端开始，逐个进行记录的关键字和给定值的比较。找到，返回该元素的位置序号，未找到，返回0。

```
int LocateELem(SqList L, ElemType e){
//在线性表L中查找值为e的数据元素，返回其序号(是第几个元素)
    for (i=0; i< L.length; i++)
        if (L.elem[i]==e) return i+ 1; //查找成功，返回序号
    return 0; //查找失败,返回0
}
```

2.2.2 顺序表基本操作的实现

【算法2.3】顺序表的查找

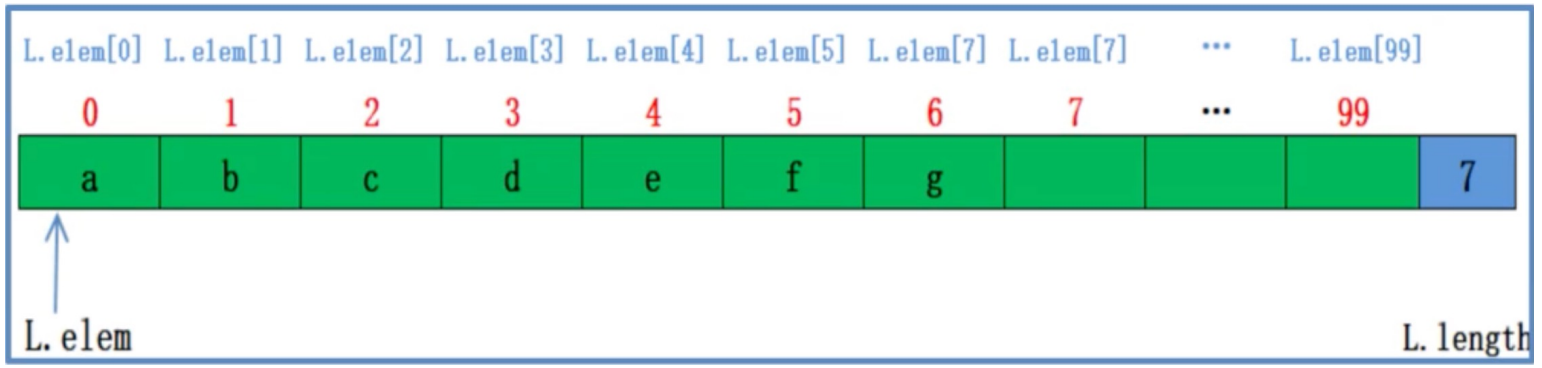
- 在线性表L中查找与指定值e相同的数据元素的位置
- 从表的一端开始，逐个进行记录的关键字和给定值的比较。找到，返回该元素的位置序号，未找到，返回0。

```
int LocateELem(SqList L, ElemType e){
//在线性表L中查找值为e的数据元素，返回其序号(是第几个元素)
    i=0;
    while(i<L.length&&L.elem[i]!=e) i++;
    if (i<L.length) return i+ 1; //查找成功，返回序号
    return 0; //查找失败,返回0
}
```

2.2.2 顺序表基本操作的实现

- 【算法2.3】顺序表的查找算法分析
- 因为查找算法的基本操作为: 将记录的关键字同给定值进行比较

  基本操作: L.elem[i]==e



比较次数: e=a, 1次; e=b, 2次; e=c, 3次; ….. , e=g, 7次;

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 【算法2.3】顺序表的查找算法分析
- 平均查找长度ASL (Average Search Length):
  - 为确定记录在表中的位置，需要与给定值进行比较的关键字的个数的期望值叫做查找算法的平均查找长度。

2.2.2 顺序表基本操作的实现

杭州电子科技大学 HANGZHOU DIANZI UNIVERSITY

【算法2.3】顺序表的查找算法分析

对含有n个记录的表，查找成功时：

$$ASL = \sum_{i=1}^{n} P_i C_i$$

C_i：找到第i个记录需比较的次数。

P_i：第i个记录被查找的概率。

顺序查找的平均查找长度:

$$ASL = P_1 + 2P_2 + \cdots + (n-1)P_{n-1} + nP_n$$

假设每个记录的查找概率相等： $P_i = \frac{1}{n}$

则：

$$ASL_{SS} = \sum_{i=1}^{n} P_i C_i = \frac{1}{n}\sum_{i=1}^{n} i = \frac{n+1}{2}$$

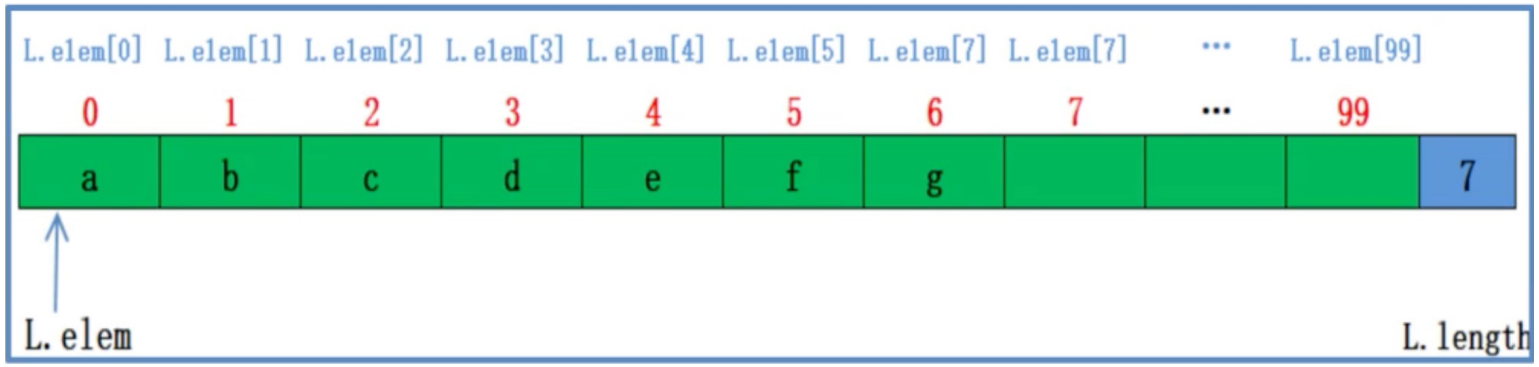
2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

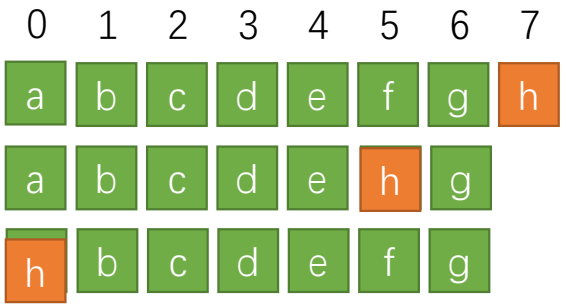
- 线性表的基本操作：
  - InitList(&L)　　//初始化操作，建立一个空的线性表L
  - DestroyList(&L)　　//销毁已存在的线性表L
  - ClearList(&L)　　//将线性表清空
  - ListInsert(&L, i, e)　　//在线性表L中第i个位置插入新元素e
  - ListDelete(&L, i, &e)　　//删除线性表L中第i个位置元素，用e返回
  - IsEmpty(L)　　//若线性表为空，返回true，否则false
  - ListLength(L)　　//返回线性表L的元素个数
  - LocateElem(L, e)　　//L中查找与给定值e相等的元素，若成功返回该元素在表中的序号，否则返回0
  - GetElem(L, i, &e)　　// 将线性表L中的第i个位置元素返回给e

2.2.2 顺序表基本操作的实现

顺序表的插入



插入不同位置的算法演示：



- 插入位置在最后
- 插入位置在中间
- 插入位置在最前面

2.2.2 顺序表基本操作的实现

顺序表的插入

线性表的插入运算是指在表的第i (1≤i ≤ n+1)个位置上，插入一个新结点e，使长度为n的线性表(a1,... ai-1, ai, ... an)变成长度为n+1的线性表(a1, ..., ai -1, e, ai, ..., an)

算法思想:

①判断插入位置i是否合法。

②判断顺序表的存储空间是否已满，若已满返回ERROR。

③将第n至第i位的元素依次向后移动一个位置，空出第i个位置。

④将要插入的新元素e放入第i个位置。

⑤表长加1，插入成功返回OK。

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【算法2.4】顺序表的插入

```
Status ListInsert_ Sq(SqList &L, int i , ElemType e){
①   if(i<1 || i> L.length+ 1) return ERROR;     //i值不合法
②   if(L.length= = MAXSIZE) return ERROR; //当前存储空间已满
③   for(j=L.length-1;j>=i-1;j--)
        L.elem[j+1]=L.elem[j];                  //插入位置及之后的元素后移
④   L.elem[i-1]=e;                              //将新元素e放入第i个位置
⑤   L.length++;                                 //表长增一
    return OK;
}
```

2.2.2 顺序表基本操作的实现

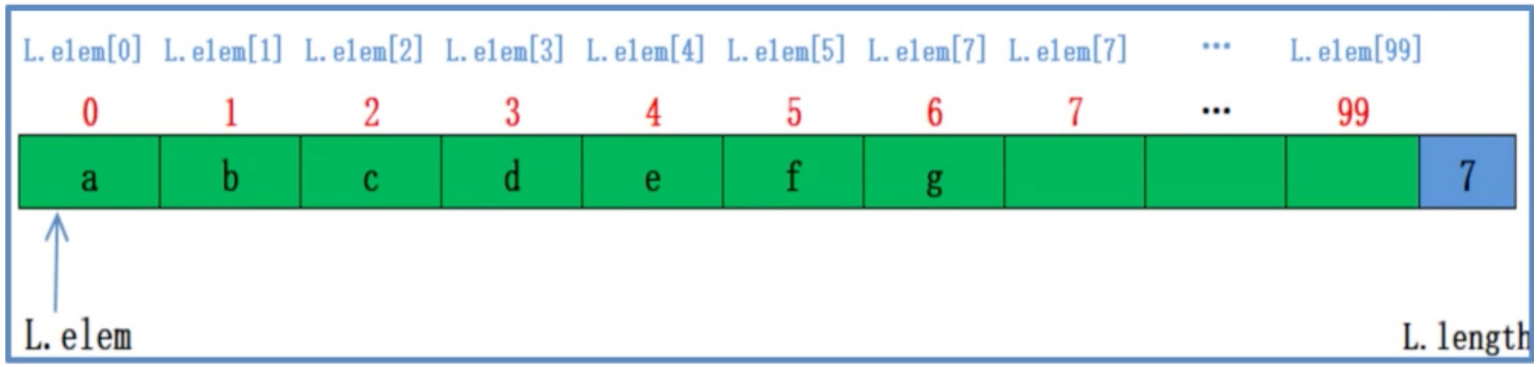
杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【算法2.4】顺序表的插入算法分析

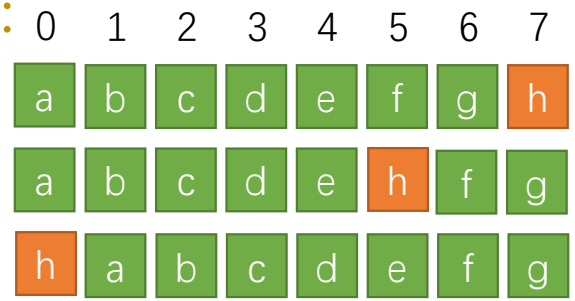
- 算法时间主要耗费在移动元素的操作上
  - 若插入在尾结点之后，则根本无需移动(特别快)；
  - 若插入在首结点之前，则表中元素全部后移(特别慢)；
  - 若要考虑在各种位置插入(共n+ 1种可能)的平均移动次数，该如何计算?

2.2.2 顺序表基本操作的实现

顺序表的插入



插入不同位置的算法演示：



- 插入位置在最后
- 插入位置在中间
- 插入位置在最前面

2.2.2 顺序表基本操作的实现

【算法2.4】顺序表的插入算法分析

- 算法时间主要耗费在移动元素的操作上
  - 若插入在尾结点之后，则根本无需移动(特别快)；
  - 若插入在首结点之前，则表中元素全部后移(特别慢)；
  - 若要考虑在各种位置插入(共n+ 1种可能)的平均移动次数，该如何计算?

$$E_{ins} = \frac{1}{n+1}\sum_{i=1}^{n+1}(n-i+1) = \frac{1}{n+1}(N+\cdots+1+0)$$

$$= \frac{1}{n+1}\frac{n(n+1)}{2} = \frac{n}{2}$$

  - 顺序表插入算法的平均时间复杂度为O(n)。

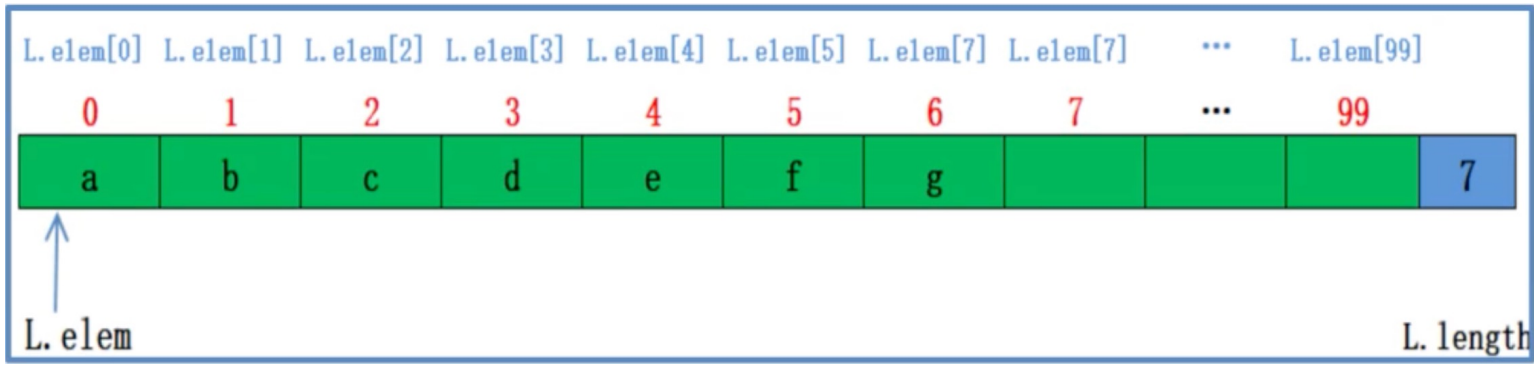
2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

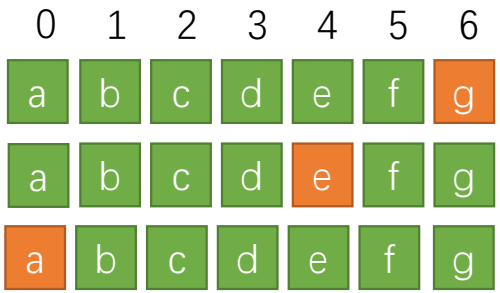
- 线性表的基本操作：
  - InitList(&L)　　//初始化操作，建立一个空的线性表L
  - DestroyList(&L)　　//销毁已存在的线性表L
  - ClearList(&L)　　//将线性表清空
  - ListInsert(&L, i, e)　　//在线性表L中第i个位置插入新元素e
  - ListDelete(&L, i, &e)　　//删除线性表L中第i个位置元素，用e返回
  - IsEmpty(L)　　//若线性表为空，返回true，否则false
  - ListLength(L)　　//返回线性表L的元素个数
  - LocateElem(L, e)　　//L中查找与给定值e相等的元素，若成功返回该元素在表中的序号，否则返回0
  - GetElem(L, i, &e)　　// 将线性表L中的第i个位置元素返回给e

2.2.2 顺序表基本操作的实现

顺序表的删除



删除算法演示：



- 删除位置在最后
- 删除位置在中间
- 删除位置在最前面

2.2.2 顺序表基本操作的实现

顺序表的删除

线性表的删除运算是指在表的第i (1≤i ≤ n)个结点删除使长度为n的线性表(a1,... ai-1, ai, ... an)变成长度为n-1的线性表(a1, ..., ai -1, ai+1, ..., an)

算法思想:

①判断插入位置i是否合法（合法值为1≤i≤n）。

②将欲删除的元素保留在e中。

③将第i+1至第n位的元素依次向前移动一个位置。

④表长减1，删除成功返回OK。

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

【算法2.5】顺序表的删除

```
Status ListDelete_Sq(SqList &L, int i, &e) {
①     if((i<1)||(i>L.length)) return ERROR;    //i值不合法
②     e = L.elem[i-1];
③     for (j=i; j<=L.length-1; j++)
          L.elem[j-1]=L.elem[j];                //被删除元素之后的元素前移
④     L.length--;                              //表长减1
    return OK;
}
```

2.2.2 顺序表基本操作的实现

【算法2.5】顺序表的删除算法分析

- 算法时间主要耗费在移动元素的操作上
  - 若删除尾结点，则根本无需移动(特别快)；
  - 若删除首结点，则表中n-1个元素全部前移(特别慢)；
  - 若要考虑在各种位置删除(共n种可能)的平均移动次数，该如何计算?

$$E_{del} = \frac{1}{n}\sum_{i=1}^{n}(n-i) = \frac{1}{n}\frac{(n-1)n}{2} = \frac{n-1}{2}$$

  - 顺序表删除算法的平均时间复杂度为O(n)。

顺序表（线性表的顺序存储结构）的特点

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

(1)利用数据元素的存储位置表示线性表中相邻数据元素之间的先后关系，即线性表的逻辑结构与存储结构一致

(2)在访问线性表时，可以快速地计算出任何一个数据元素的存储地址。因此可以粗略地认为，访问每个元素所花时间相等

- 这种存取元素的方法被称为随机存取法

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 线性表的基本操作：
  - InitList(&L)　　//初始化操作，建立一个空的线性表L
  - DestroyList(&L)　　//销毁已存在的线性表L
  - ClearList(&L)　　//将线性表清空
  - ListInsert(&L, i, e)　　//在线性表L中第i个位置插入新元素e
  - ListDelete(&L, i, &e)　　//删除线性表L中第i个位置元素，用e返回
  - IsEmpty(L)　　//若线性表为空，返回true，否则false
  - ListLength(L)　　//返回线性表L的元素个数
  - LocateElem(L, e)　　//L中查找与给定值e相等的元素，若成功返回该元素在表中的序号，否则返回0
  - GetElem(L, i, &e)　　// 将线性表L中的第i个位置元素返回给e

2.2.2 顺序表基本操作的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

顺序表的操作算法分析

- 时间复杂度
  - 查找、插入、删除算法的平均时间复杂度为O(n)
- 空间复杂度
  - 显然，顺序表操作算法的空间复杂度S(n)=O(1)

    (没有占用辅助空间)

顺序表优缺点

- 优点
  - 存储密度大(结点本身所占存储量/结点结构所占存储量)
  - 可以随机存取表中任一元素

- 缺点
  - 在插入、删除某一元素时，需要移动大量元素
  - 浪费存储空间
  - 属于静态存储形式，数据元素的个数不能自由扩充

克服这一缺点 → 链表

课后题

【题】设顺序表va中的数据元素非递减有序。试写一算法，将x 插入到顺序表的适当位置上，以保持该表的有序性。

课后题


```
Status InsertOrderList (SqList &va, ElemType x){
//在非递减的顺序表va 中插入元素x 并使其仍成为有序顺序表的算法
    int i;
    if(va.length==va.listsize) return(OVERFLOW);
    for(i=va.length; i>0, x<va.elem[i-1]; i--)
        va.elem[i]=va.elem[i-1];
    va.elem[i]=x;
    va.length++;
    return OK;
}
```

课后题

【题】假设以两个元素依值递增有序排列的线性表A 和B 分别表示两个集合（即同一表中的元素值各不相同），现要求另辟空间构成一个线性表C，其元素为A 和B 中元素的交集，且表C 中的元素有依值递增有序排列。试对顺序表编写求C 的算法。

课后题


```
// 将A、B 求交后的结果放在C 表中
Status ListCross_Sq (SqList &A, SqList &B, SqList &C){
        int i=0, j=0, k=0;
        while(i<A.length && j<B.length){
                if(A.elem[i]<B.elem[j]) i++;
                else
                        if(A.elem[i]>B.elem[j]) j++;
                        else{
                                ListInsert_Sq(C,k,A.elem[i]);
                                i++;
                                k++;
                         }
         }
         return OK;
}
```